gSpan Algorithm

Thursday, 30th May, 2019
Deepti Saravanan

REFERENCE.

gSpan: Graph-Based Substructure Pattern Mining by X. Yan and J. Han

Context

- Limitations of Apriori
- Problem Statement
- DFS Lexicographic Order - Steps
- DFS Subscripting
- Forward and Backward Edges - Rules
- DFS Code
- DFS Lexicographic Order - Rules
- Minimum DFS Code
- DFS Code Tree
- gSpan Algorithm
- Evaluation

Limitations of Apriori

1. Candidate generation: The generation of size (k + 1) sub- graph candidates from size k frequent subgraphs is more complicated and costly.
2. Pruning false positives: Subgraph isomorphism test is an NP- complete problem; thus pruning false positives is costly.

Problem Statement

To avoid the significant losses of the Apriori algorithm.

- gSpan introduced lexicographic ordering and formation of DFS Tree using DFS Code.
- Combines the growing and checking of frequent subgraphs into one procedure, thus accelerating the mining process.

DFS Lexicographic Order

Steps include:

- DFS Subscripting
- DFS Code
- DFS Lexicographic Order
- Minimum DFS Code
- DFS Code Tree

Support denotes the number of graphs in which the subgraph occurs.

DFS Subscripting

Subscripting is used to label the order of the different DFS trees obtained from the isomers of the given graph according to their discovery times, that is $i<j$ implies v_i is discovered before v_j.

Isomorphism of graphs - Edge-preserving bijection. It is the bijection between the two sets of vertices of the graphs respectively.

Let v_0 be the root vertex and v_n be the final vertex. **Rightmost path** is defined as the straight path from v_0 to v_n.

Let the subscripted G be G_T.

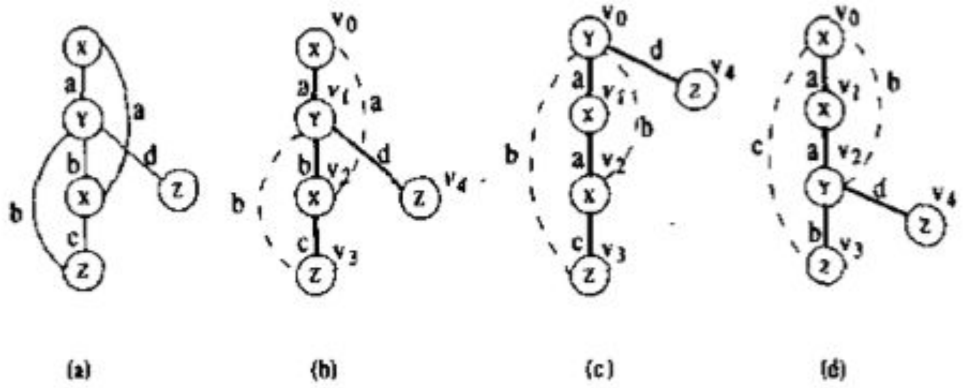

Figure 1. Depth-First Search Tree

Forward and Backward Edge

Forward edge - All the edges in the DFS tree.

Backward edge - All the edges not in the DFS tree.

Let (i,j) be the ordered pair representing an edge.

if(i<j) -> forward edge

Else -> backward edge

Rules

A linear order ($<_T$) is built for all the edges in G based on 3 rules:

Let $e_1 = (i_1,j_1)$ and $e_2 = (i_2,j_2)$.

1. If $i_1=i_2$ and $j_1<j_2$, then $e_1 <_T e_2$.
2. If $i_1<j_1$ and $j_1=i_2$, then $e_1 <_T e_2$.
3. If $e_1 <_T e_2$ and $e_2 <_T e_3$, then $e_1 <_T e_3$.

DFS Code - code(G,T)

Given a DFS free T for a graph G, an edge sequence (e_i) can be constructed based on $<_T$, such that $e_i <_T e_{i+1}$, where i = 0,. ..,|E|- 1.

It is presented as a 5 tuple $(i,j,l_i,l_{(i,j)},l_j)$ where l_i and l_j are the labels of the vertices v_i and v_j respectively and $l_{(i,j)}$ is the label of the edge between them.

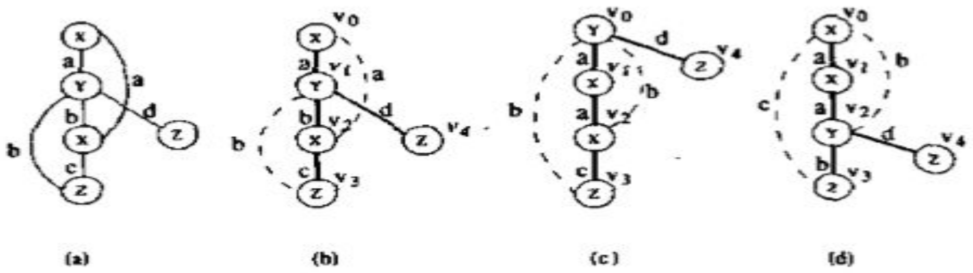

Figure 1. Depth-First Search Tree

edge	(Fig 1b) α	(Fig 1c) β	(Fig 1d) γ
0	$(0, 1, X, a, Y)$	$(0, 1, Y, a, X)$	$(0, 1, X, a, X)$
1	$(1, 2, Y, b, X)$	$(1, 2, X, a, X)$	$(1, 2, X, a, Y)$
2	$(2, 0, X, a, X)$	$(2, 0, X, b, Y)$	$(2, 0, Y, b, X)$
3	$(2, 3, X, c, Z)$	$(2, 3, X, c, Z)$	$(2, 3, Y, b, Z)$
4	$(3, 1, Z, b, Y)$	$(3, 0, Z, b, Y)$	$(3, 0, Z, c, X)$
5	$(1, 4, Y, d, Z)$	$(0, 4, Y, d, Z)$	$(2, 4, Y, d, Z)$

Table 1. DFS codes for Fig. 1(b)-(d)

DFS Lexicographic Order

Z - set of all DFS Codes for all the connected graphs.

$<_L$ - Linear order in the label set L.

$<_e$ - Lexicographic combination of $<_T$ and $<_L$.

If a = code(G_a,T_a) = (a_0,a_1,...,a_m) and b = code(G_b,T_b) = (b_0,b_1,...,b_m), and a,b in Z, then a<=b iff either of the following conditions is true.

1. For every t, 0 < t < min(m,n), a_h = b_h, for k < t, $a_t <_e b_t$.
2. a_k = b_k for 0<=k<=m and n>=m.

$a_t <_e b_t$ condition is true if either of the following conditions is true:

1. $x_t \in E^b_{\mathbf{x}}$, and $y_t \in E^f_{\mathbf{y}}$.
2. $x_t \in E^b_{\mathbf{x}}$, $y_t \in E^b_{\mathbf{y}}$, and $j_x < j_y$.
3. $x_t \in E^b_{\mathbf{x}}$, $y_t \in E^b_{\mathbf{y}}$, $j_x = j_y$, and $l_{(i_x,j_x)} \prec_L l_{(i_y,j_y)}$.
4. $x_t \in E^b_{\mathbf{x}}$, $y_t \in E^b_{\mathbf{y}}$, $j_x = j_y$, $l_{(i_x,j_x)} = l_{(i_y,j_y)}$, and $d_{(i_x,j_x)} \prec_D d_{(i_y,j_y)}$.
5. $x_t \in E^f_{\mathbf{x}}$, $y_t \in E^f_{\mathbf{y}}$, and $i_y < i_x$.
6. $x_t \in E^f_{\mathbf{x}}$, $y_t \in E^f_{\mathbf{y}}$, $i_x = i_y$, and $l_{i_x} < l_{i_y}$.
7. $x_t \in E^f_{\mathbf{x}}$, $y_t \in E^f_{\mathbf{y}}$, $i_x = i_y$, $l_{i_x} = l_{i_y}$, and $l_{(i_x,j_x)} \prec_L l_{(i_y,j_y)}$.
8. $x_t \in E^f_{\mathbf{x}}$, $y_t \in E^f_{\mathbf{y}}$, $i_x = i_y$, $l_{i_x} = l_{i_y}$, $l_{(i_x,j_x)} = l_{(i_y,j_y)}$, and $l_{j_x} \prec_L l_{j_y}$.
9. $x_t \in E^f_{\mathbf{x}}$, $y_t \in E^f_{\mathbf{y}}$, $i_x = i_y$, $l_{i_x} = l_{i_y}$, $l_{(i_x,j_x)} = l_{(i_y,j_y)}$, $l_{j_x} = l_{j_y}$, and $d_{(i_x,j_x)} \prec_D d_{(i_y,j_y)}$.

Minimum DFS Code

Given a G and Z(G), the minimum Z(G) according to the lexicographic order is called the minimum DFS Code/Canonical Label of G.

THEOREM. Given two graphs G and G', G is isomorphic to G' iff min(G) = min(G').

- Frequent subgraph mining equivalent to mining their corresponding minimum DFS codes.
- Parent produces child by growing an edge only from the vertices on the rightmost path.
- While forward edges can be grown from any of the vertices in the rightmost path, backward edges can only be grown from the rightmost vertex.

DFS Code Tree

- Each node represents a DFS Code.
- Relationship between siblings is in accordance with the lexicographic order.
- Pre-order traversal of DFS Code Tree follows the lexicographic order.
- It is a finite tree as only subgraphs above the minimum support value is included.
- Starting from zero edges, the node produces children by increasing the edge count by one in every level. The support of that subgraph is calculated. If it is above the min_sup subgraph count, it is considered for further process of producing children. If not, the pathway is pruned.

DFS Code Tree (contd.)

- This is done to make sure that the final subgraphs obtained has support value above min_sup, and also it should comprise of only those subgraphs whose support values are also above min_sup.
- Upon reaching the nth level (n-1) edges, through depth-first search of the code tree, all the minimum DFS codes of frequent subgraphs can be discovered.
- In the case where two nodes, say s and s, containing same graph but differeny DFS codes$'$, then it can be implied that s$'$ does not contain any minimum DFS code and hence can be pruned. This is in accordance to the fact that a subgraph can have only one min DFS Code.

gSpan Algorithm

Uses Sparse Adjacency List to store graphs.

Graphset_projection(D,S)

- ❏ D(Graph Dataset) is sorted by their frequency.
- ❏ Remove the infrequent vertices and edges.
- ❏ Relabel the remaining vertices and edges.
- ❏ S' - all the frequent 1-edge graphs in D.
- ❏ Store it in S(contains the mining result).
- ❏ Sort S' in DFS Lexicographic Order.

- For each edge(initialised to s) in S', consider only the graphs which has that particular edge(when the subgraph size is large).
- Call the subgraph_mining subprocedure
- Once it returns the final s, remove the edge initially considered from D(shrinking the dataset).
- If $|D| < min\_sup$, break. Or else, repeat the algorithm for the remaining edges.

Subprocedure - subgraph_mining

- ❏ Check if s is the minimum DFS code. Further steps are continued only if this is true.
- ❏ Enumerate s in each graph in D and count its children.
- ❏ For each child, compute the support value and check if it is more than the min_sup.
- ❏ If yes, then assign it to s and repeat the subgraph_mining subprocedure.
- ❏ Else, return s to the caller.

Evaluation

In both synthetic and chemical compound dataset, the gspan algorithm out-performed the FSG. The comparison was made based on the runtime of the two algorithms.

THANK YOU